Systems Software Services Technical Bulletin

Number:	0026
Issued Date:	February 22, 1983
Effective Date:	
Section/Groups:	
Submitted By:	
Approved By:	

Operational Tape Management

The purpose of this seminar is to discuss and illustrate ways of handling tapes efficiently. Many benefits of good tape usage can be realized. to mention a few:

- Considerable operator time may be saved by avoiding repeated mounts of the same tape. This seems obvious, but it does not occur.

- When several tapes are required in sequence for a job, it is desirable to request each tape mount on the same drive. This will probably avoid mounting a tape on the wrong drive and make the operator's work much easier.

- Computer throughput can be improved by freeing the tape drive for other jobs when you no longer need it.

- Frequently a tape is needed by several steps of a job. Operator intervention can be avoided by passing the tape from step to step.

- Many jobs use tapes conditionally. Deferred mounting should be used in these cases to avoid mounting tapes that may not be needed.

I. The Overworked Operator

Following are two examples that demonstrate the improper and proper way to handle multi-file tapes. The job will copy a partitioned data set from disk to tape as file #1, a sequential data set from disk to tape is file #2, and a partitioned data set from disk to tape as file #3. The intention of the programmer creating the job stream is good because he wants to put all three data sets on one reel of tape rather than creating three separate tapes, but watch what happens to the tape (and the operator) when we run this job!

(The Wrong Way)

```
//STEP1         EXEC PGM=IEBCOPY
//SYSPRINT DD        SYSOUT=*
//DISK          DD      DSN=GP.EXAMPLE.A.PO,DISP=(OLD,KEEP)
//TAPE          DD      DSN=GP.FILE1.BKUP,
//                      UNIT=TAPE,
//                      VOLUME=SER=OS1234,
//                      DISP=(NEW,KEEP),
//                      LABEL=(1,SL,EXPDT=98010)
//SYSIN                 DD *
 COPY 0=TAPE,I=DISK
/*
//*******************************************************
//STEP2         EXEC  PGM=IEBGENER
//SYSPRINT      DD      SYSOUT=*
//SYSUT1        DD      DSN=GP.EXAMPLE.B.PS,DISP=(OLD,KEEP)
//SYSUT2        DD      DSN=GP.FILE2.BKUP,
//                      UNIT=TAPE,
//                      VOLUME=SER=OS1234,
//                      DISP=(NEW,KEEP),
//                      LABEL=(2,SL,EXPDT=98010)
//SYSIN         DD      DUMMY
//********************************************************
//STEP3         EXEC  PGM=IEBCOPY
//SYSPRINT      DD      SYSOUT=*
//DISK          DD      DSN=GP.EXAMPLE.C.PO,DISP=(OLD,KEEP)
//TAPE          DD      DSN=GP.FILE3.BKUP,
//                      UNIT=TAPE,
//                      VOLUME=SER=OS1234,
//                      DISP=(NEW,KEEP),
//                      LABEL=(3,SL,EXPDT=98010)
//SYSIN                 DD *
 COPY 0=TAPE,I=DISK
/*
```

Notice that each step rewinds and unloads the tape and that the operator must remount the tape (on a different tape drive) for each step. Notice also that following each mount, the system must reread over the previous data to properly position the tape for the next file. This condition can be corrected by using "PASS" instead of "KEEP". The use of the "RETAIN" parameter is not necessary for positioning reasons, but may help avoid a unit swap under certain conditions.

(The Right Way)

```
//STEP1         EXEC PGM=IEBCOPY
//SYSPRINT DD        SYSOUT=*
//DISK          DD   DSN=GP.EXAMPLE.A.PO,DISP=(OLD,KEEP)
//TAPE          DD   DSN=GP.FILE1.BKUP,
//                   UNIT=TAPE,
//                   VOLUME=(,RETAIN),
//                   DISP=(NEW,PASS),                ***This is the key!!***
//                   LABEL=(1,SL,EXPDT=98010)
//SYSIN              DD *
 COPY 0=TAPE,I=DISK
/*
//*****************************************************
//STEP2         EXEC PGM=IEBGENER
//SYSPRINT      DD   SYSOUT=*
//SYSUT1        DD   DSN=GP.EXAMPLE.B.PS,DISP=(OLD,KEEP)
//SYSUT2        DD   DSN=GP.FILE2.BKUP,
//                   UNIT=TAPE,
//                   VOLUME=(,RETAIN,REF=*.STEP1.TAPE),
//                   DISP=(NEW,PASS),
//                   LABEL=(2,SL,EXPDT=98010)
//SYSIN         DD   DUMMY
//*****************************************************
//STEP3         EXEC PGM=IEBCOPY
//SYSPRINT      DD   SYSOUT=*
//DISK          DD   DSN=GP.EXAMPLE.C.PO,DISP=(OLD,KEEP)
//TAPE          DD   DSN=GP.FILE3.BKUP,
//                   UNIT=TAPE,
//                   VOLUME=REF=*.STEP2.TAPE,
//                   FREE=CLOSE,
//                   DISP=(NEW,KEEP),
//                   LABEL=(3,SL,EXPDT=98010)
//SYSIN              DD *
 COPY 0=TAPE,I=DISK
/*
```

Notice that the tape does not rewind at the end of each step, each succeeding file is placed immediately after the previous file, and no operator intervention is required.

II. My Job Won’t Run

When all of the available tape drives have been allocated to various jobs, no new jobs that require tapes can be run. There are some steps that programmers can take to help in this regard.

If your job step uses several tapes sequentially, such as input to a sort, your job will run more smoothly if all input tapes can be mounted on the same drive. Here are a couple of examples:

(The Wrong Way)

```
//SORT       EXEC PGM=SORT
//SYSOUT     DD   SYSOUT=*
//SORTIN     DD   DSN=GP.SORTIN1,DISP=(OLD,KEEP)
//           DD   DSN=GP.SORTIN2,DISP=(OLD,KEEP)
//           DD   DSN=GP.SORTIN3,DISP=(OLD,KEEP)
//SORTWK01 DD     UNIT=SYSDA,SPACE=(CYL,10)
//SORTWK02 DD     UNIT=SYSDA,SPACE=(CYL,10)
//SORTWK03 DD     UNIT=SYSDA,SPACE=(CYL,10)
//SORTOUT    DD   DSN=GP.SORTOUT,
//                DISP=(NEW,KEEP),
//                UNIT=TAPE,
//                LABEL=(1,SL,EXPDT=98003)
//SORTIN          DD    *
 SORT FIELDS=(1,10,A),FORMAT=BI
/*
```

(The Right Way)

```
//SORT       EXEC PGM=SORT
//SYSOUT     DD   SYSOUT=*
//SORTIN     DD   DSN=GP.SORTIN1,DISP=(OLD,KEEP)
//           DD   DSN=GP.SORTIN2,DISP=(OLD,KEEP),UNIT=AFF=SORTIN
//           DD   DSN=GP.SORTIN3,DISP=(OLD,KEEP),UNIT=AFF=SORTIN,
//                FREE=CLOSE
//SORTWK01 DD     UNIT=SYSDA,SPACE=(CYL,10)
//SORTWK02 DD     UNIT=SYSDA,SPACE=(CYL,10)
//SORTWK03 DD     UNIT=SYSDA,SPACE=(CYL,10)
//SORTOUT    DD   DSN=GP.SORTOUT,
//                DISP=(NEW,KEEP),
//                UNIT=TAPE,
//                FREE=CLOSE,
//                LABEL=(1,SL,EXPDT=98003)
//SORTIN          DD    *
 SORT FIELDS=(1,10,A),FORMAT=BI
/*
```

By using unit affinity, the tapes will be mounted on the same drive one after another, thus avoiding the allocation of several drives.

When coding an application program (Fortran, Cobol, Bal) it is a common tendency to open all files at the beginning of the program with a single "OPEN" statement and to close all files at the end of the program with a single "CLOSE" statement. When a file is opened, the tape must be mounted immediately even if the file will not be read until several hours later. It would be better if the programmer would open the file when it is needed rather than open it at the beginning of the program for the sake of programming convenience. In this situation, one tape drive will still be allocated, even if not used. When closing files at the end of the program, it is possible for the tape to remain idle after use for many hours of processing when it could have been closed by the program and unloaded with the "FREE" parameter. Consider an application program that uses two input tapes. It is desired to process the first file, unload the tape, and process the second file, preferably on the same tape drive.

The DD Cards

```
//MSTR1     DD    DSN=FILE1,
//                DISP=(OLD,KEEP),
//                VOL=SER=OSAAAA,
//                FREE=CLOSE,
//                UNIT=(TAPE,,DEFER)
//MSTR2     DD    DSN=FILE2,
//                DISP=(OLD,KEEP),
//                VOL=SER=OSBBBB,
//                FREE=CLOSE,
//                UNIT=AFF=MSTR1
```

will accomplish all of the following:

If you wish to process both tapes, the sequence of event will be:

1. When the application program tries to open the first file, a mount instruction will be issued to the operator to mount the first tape.

2. The program will open, process, and close the first tape.

3. The first tape will be unloaded (caused by using the "FREE" parameter).

4. When the application program tries to open the second file, a mount instruction will be issued to the operator to mount the second tape (on the same drive from which the first one was removed). This is caused by coding "UNIT=AFF=MSTR1" which also achieves the benefit of deferred mounting.

5. The program will open, process, and close the second tape.

6. The second tape will be unloaded (caused by using the "FREE" parameter). This single event will free up the tape drive for use by other jobs, even though your job step continues to process for many hours.

If your program logic does not allow the second tape to be used, then the second tape will never be mounted (remember, "UNIT=AFF=MSTR1" implies "DEFER").

If your program logic does not allow the first tape to be used, then the first tape will never be mounted because of the "DEFER" parameter on the first DD statement.

If your program logic does not allow either tape to be used, then neither tape will be mounted.

If you wish to avoid allocation of a drive, you may code "DUMMY" instead of "DSNAME", if you are sure that the file will never be used.

When you have several volumes of tape for a single input file, please consider the effect of mounting several volumes concurrently, particularly if the job will run for a long time:

```
//MULTIVOL DD    DSN=GP.MASTER,
//               UNIT=(TAPE,8),          **This is being too greedy!!**
//               DISP=(OLD,KEEP),
//               VOLUME=SER=(OSAAAA,OSBBBB,OSCCCC,OSDDDD,OSEEEE,
//               OSFFFF,OSGGGG,OSHHHH),
/./              FREE=CLOSE
```

A job step using this DD card would probably never execute because it would not be able to get all eight tape drives for itself; however, if it did execute, no other tape oriented jobs could execute because they could not find any tape drives either. A better DD card might be:

```
//MULTIVOL DD    DSN=GP.MASTER,
//               UNIT=(TAPE,2),          **This is about right!!**
//               DISP=(OLD,KEEP),
//               VOLUME=SER=(OSAAAA,OSBBBB,OSCCCC,OSDDDD,OSEEEE,
//               OSFFFF,OSGGGG,OSHHHH),
/./              FREE=CLOSE
```

Mounting two tapes concurrently will not overallocate drives while achieving the benefit of continuous tape drive operation, i.e., there will always be a tape mounted for processing while the operator is finding and mounting the next tape.

III. My Job Runs Ooooohh So Slow

How many times have you had to stay late waiting for a slow job to run? If it is a tape oriented job, you might want to consider whether or not you have the files blocked efficiently. Note the following statistics regarding IBM 6250 BPI tape drives:

> A tape blocked at 4000 will run 2.0 times faster than one blocked at 1000.
> A tape blocked at 8000 will run 2.5 times faster than one blocked at 1000.
> A tape blocked at 4000 will run 17.0 times faster than one blocked at 80.
> A tape blocked at 8000 will run 21.0 times faster than one blocked at 80.

If you had a file that would process in 15 minutes if blocked at 8000, but you blocked it at 80 instead, you job would run for 5 hours and 15 minutes!

The ideal block size for tape data sets is 8000 to 32000.

IV. Ooooh They scratche My Tape *7(6$**

No, we didn't scratch your tape. Sorry, you scratched your own tape! Actually the scratching of tapes is controlled by the Tape Management System (TMS) which is controlled by your Job Control Language (JCL). We would like to encourage all of our customers to be very familiar with the use of the Tape Management System. Also be aware that there is on online inquiry function available called TIQ. We also encourage complete familiarity with OS JCL including generation data groups and all other aspects of tape management.

V. My Job Created 62 Tapes Last Month!

Are you running a job every night that captures the day's work and writes it on a tape and a copy to a backup tape? This is a frequently encountered situation. Some examples might be: SMF machine job statistics, CICSS analyzer data, checks for Social Services recipients, etc. Tapes are created each night and saved for the entire month so that reports can be produced at the end of the month. As you can see, it doesn't take long to create a library full of tapes and consequently a tape shrotage soon occurs.

There is another way:

```
//ONCEADAY      DD    DSN=GP.BILLING.JAN83,
//              DISP=(MOD,CATLG),
//              FREE=CLOSE,
//              UNIT=(TAPE,,DEFER),
//              LABEL=(1,SL,EXPDT=?????),
//              DCB=(RECFM=??,LRECL=????,BLKSIZE=?????,DEN=?)
```

This DD card will allow you to extend (MODify) the file each time the job is run. The tape is simply read up to the end-of-file mark then the new day's data is written over and past the end-of-file mark and a new end-of-file makr is written at the end of the data. One should be sure to process a backup copy when operating in this mode since there is a slightly higher risk of data loss (more tape handling for the individual tapes, and loss of a tape means loss of the month, not just a single day). Another drawback is that the tapes must be reread to the end-of-file mark each time they are processed resulting in a fair amount of redundant tape read processing. It is generally believed that this method is a fair tradeoff against creating 62 tapes each month and the difficulty associated with processing 31 of them at a time to produce a monthly tape and/or report.

Notice the "CATLG" parameter on an existing cataloged file—seems redundant. Actually it is every time you run it except when the program calls for a new reel of tape which needs to be cataloged! Same remarks apply to "MOD". You will get a new tape label in the JES listing each time the program is run. If a new reel of tape is created, you may want to keep this label for reference. Of course all of the information is available in the TMS catalog. Similar remarks apply to the "LABEL" and "DCB" parameters.

If you choose this method, it will be necessary to initialize your tapes at the beginning of each month:

```
//RESETMOD EXEC PGM=IEBGENER     ***INITIALIZE NEW TAPE FOR NEW
                                 MONTH***
//SYSPRINT  DD    SYSOUT=*
//SYSUT1    DD    DUMMY,         ***PUTS EOF AT BEGINNING OF THE
                                 TAPE***
//                DCB=(RECFM=??,LRECL=????,BLKSIZE=?????,DEN=?)
//SYSUT2    DD    DSN=GP,BILLING,JAN83,
//                DISP=(NEW,CATLG,DELETE),
//                UNIT=TAPE,
//                LABEL=(1,SL,EXPDT=?????),
//                DCB=(RECFM=??,LRECL=????,BLKSIZE=?????,DEN=?)
//SYSIN     DD    DUMMY
```

VI. For More Information

C S.I.S.C. Tecnical Bulletin #8, February 10, 1982 (Job Performance Tips)

C MVS Job Control Language

C UCC1 Tape Management System (TMS) (User’s Reference Manual)